HOSTED BY
ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jides

Wavelet decomposition of software entropy reveals symptoms of malicious code

*Michael Wojnowicz**, *Glenn Chisholm*, *Matt Wolff*, *Xuan Zhao*

Cylance, Inc., 18201 Von Karman Ave., Irvine, CA 92612, United States

HIGHLIGHTS

- Development of new features for machine learning.
- Application of wavelet transforms to software entropy.
- Discovery of suspicious patterns of entropic change.
- Automatic classification of parasitic malware.

ARTICLE INFO

Article history:
Received 21 July 2016
Accepted 31 October 2016
Published online 5 December 2016

Keywords:
Wavelet decomposition
Structural entropy
Malware detection
Parasitic malware
Machine learning

ABSTRACT

Sophisticated malware authors can sneak hidden malicious contents into portable executable files, and this contents can be hard to detect, especially if encrypted or compressed. However, when an executable file switches between contents regimes (e.g., native, encrypted, compressed, text, and padding), there are corresponding shifts in the file's representation as an entropy signal. In this paper, we develop a method for automatically quantifying the extent to which patterned variations in a file's entropy signal make it "suspicious". In Experiment 1, we use wavelet transforms to define a Suspiciously Structured Entropic Change Score (SSECS), a scalar feature that quantifies the suspiciousness of a file based on its distribution of entropic energy across multiple levels of spatial resolution. Based on this single feature, it was possible to raise predictive accuracy on a malware detection task from 50.0% to 68.7%, even though the single feature was applied to a heterogeneous corpus of malware discovered "in the wild". In Experiment 2, we describe how wavelet-based decompositions of software entropy can be applied to a parasitic malware detection task involving large numbers of samples and features. By extracting only string and entropy features (with wavelet decompositions) from software samples, we are able to obtain almost 99% detection of parasitic malware with fewer than 1% false positives on good files. Moreover, the addition of wavelet-based features uniformly improved detection performance across plausible false positive rates, both in a strings-only model (e.g., from 80.90% to 82.97%) and a strings-plus-entropy model (e.g. from 92.10% to 94.74%, and from 98.63% to 98.90%). Overall, wavelet decomposition of software entropy can be useful for machine learning models for detecting malware based on extracting millions of features from executable files.

Peer review under responsibility of Qassim University.
* *Corresponding author.*
E-mail address: mwojnowicz@cylance.com (M. Wojnowicz).

http://dx.doi.org/10.1016/j.jides.2016.10.009

1. Introduction

1.1. The entropy of malicious software

A fundamental goal in the information security industry is malware detection. In this paper, we focus our malware detection efforts on the fact that malicious files (e.g. parasitics, or exploits with injected shellcode) commonly contain encrypted or compressed ("packed") segments which conceal malicious contents [1]. Thus, the information security industry has been interested in developing methodologies which can automatically detect the presence of encrypted or compressed segments hidden within portable executable files. To this end, entropy analysis has been used, because files with high entropy are relatively likely to have encrypted or compressed sections inside them [2]. In general, the entropy of a random variable reflects the amount of uncertainty (or lack of knowledge) about that variable. In the context of software analysis, zero entropy would mean that the same character was repeated over and over (as might occur in a "padded" chunk of code), and maximum entropy would mean that a chunk consisted of entirely distinct values. Thus, chunks of code that have been compressed or encrypted tend to have higher entropy than native code. For instance, in the software corpus studied by [2], plain text had an average entropy of 4.34, native executables had an average entropy of 5.09, packed executables had an average entropy of 6.80, and encrypted executables had an average entropy of 7.17.

1.2. Suspiciously structured entropy

Based on the reasoning above, previous research has used high mean entropy as an indicator of encryption or compression. However, malicious contents, when concealed in a sophisticated manner, may not be detectable through simple entropy statistics, such as mean file entropy. Malware writers sometimes try to conceal hidden encrypted or compressed contents that they introduce in creating files such as parasitic malware; for instance, they may add additional padding (zero entropy chunks), so that the file passes through high entropy filters. However, files with concealed encrypted or compressed segments tend to vacillate markedly between native contents, encrypted and compressed segments, and padding, with each segment having distinct and characteristic expected entropy levels. Thus, the field of cybersecurity has started to pay attention to files with *highly structured entropy* [3,4], that is, files whose contents flips between various distinguishing levels of entropy through the file.

In order to automatically identify the degree of entropic structure within a piece of software, we represent each portable executable file as an "entropy stream." The entropy stream describes the amount of entropy over a small snippet of code in a certain location of the file. The "amount" of entropic structure can then be quantified, such that we can differentiate, for example, between a low-structured signal with a single local mean and variation around that mean, versus a highly-structured signal whose local mean changes many times over the course of the file.

In this paper,[1] we define *suspiciously structured entropy* as a *particular pattern of entropic structure* which matches those of malicious files. To quantify the suspiciousness of the structured entropy within a piece of software, we develop the notion of a "Suspiciously Structured Entropic Change Score" (SSECS). We first describe how to calculate SSECS as a single predictive feature, and analyze its performance in malware detection. We then generalize this feature to large-scale malware detection tasks. The derivation of the SSECS feature depends upon the notion of a wavelet transform, which we now briefly review.

1.3. Brief overview of wavelets

The Wavelet Transform is the primary mathematical operator underlying our quantification of structurally suspicious entropy. The Wavelet Transform extracts the amount of "detail" exhibited within a signal at various locations over various levels of resolution [7]. In essence, it transforms a one-dimensional function of "location" (in our case, file location) into a two-dimensional function of "location" and "scale." By using the output of the wavelet transform (the so-called "wavelet coefficients"), it is possible to obtain a series of coarse-to-fine approximations of an original function. These successive approximations allow us to determine the multi-scale structure of the entropy signal, in particular the "energy" available at different levels of resolution.

For this paper, we apply Haar Wavelets, which is a particularly simple family of wavelets whose members are piecewise constant. The Haar Wavelet Transform projects the original entropy signal onto a collection of piecewise constant functions which oscillates as a square wave over bounded support (i.e., these functions assume non-zero values only on certain bounded intervals). Since these piecewise constant functions have supports which vary in their scale (width) and location, the resulting projections describe the "detail" within the signal at various locations and resolutions.

More specifically, the Haar Wavelet Transform is based upon the so called "mother function", $\psi(t)$, defined by:

$$\psi(t) = \begin{cases} 1, & t \in [0, 1/2) \\ -1, & t \in [1/2, 1) \\ 0, & \text{otherwise} \end{cases}$$

a very simple step function. Given the Haar mother function $\psi(t)$, a collection of dyadically scaled and translated wavelet functions $\psi_{j,k}(t)$ are formed by:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{1}$$

where the integers j, k are scaling parameters. The dilation parameter j indexes the level of detail or spatial resolution, and the translation parameter k selects a certain location within the signal to be analyzed. Note that as the scaling parameter j increases, the function $\psi_{j,k}$ applies to (is non-zero over) successively finer intervals of the signal. Some example Haar wavelet functions are shown in Fig. 1.

[1] This paper is a development of earlier research originally published in conference proceedings [5]. For an even more comprehensive viewpoint, see [6].